Assembly and Integration Case Study: Enterprise Patch Management

Steven Lavenhar, Cigital, Inc. [vita[3]]

2007-03-19 L3 / L, M, (E)[4]

Successfully managing the inevitable changes to an enterprise-wide application is a key aspect of assembly and integration. “If these changes aren't properly managed across platforms and throughout each of the stages of the software development life cycle, production failures, including security problems, can result.” This document presents a case study of a Fortune 500 company where deficiencies in patch management left many of the company’s servers vulnerable to cyber attack and subsequent infection by the Slammer worm.

Introduction

Enterprise-wide application development projects can be extremely difficult to manage. Changes to the design specifications, documentation, and code are frequently requested by the development and testing teams, as well as end users. If these changes aren't properly managed across platforms and throughout each of the stages of the software development life cycle, production failures, including security problems, can result.

This case study is about a Fortune 500 company that developed a portal based on Plumtree Corporate Portal and several other commercial off-the-shelf (COTS) applications. Deficiencies in the management of patches resulted in the introduction of a security vulnerability that used a buffer overflow to exploit a flaw in Microsoft SQL Server 2000. This resulted in a large number of servers being infected by the Slammer worm.

The extensive use of third-party components makes it especially important to have an effective patch management system, but patch management is a reactive approach to software security that cleans up problems after they have been discovered in the field. It is always better to build security in, even though patches will sometimes be necessary.

Portal Architecture

The system architecture of the portal was designed for high availability, scalability, and high performance. The infrastructure was based on a classical n-tier design. All of the servers were configured for high availability, with no hardware single points of failure. A security-policy-based DMZ was implemented using redundant Cisco PIX firewalls. Load balancing of TCP/IP traffic across multiple Web servers was provided by redundant Cisco LocalDirectors. Clustering of application and database servers was provided by Windows 2000 Cluster Service using an active-passive configuration. Separate environments were implemented for development, testing, staging, and production. In addition, a separate environment for administration of the portal servers was utilized. The following applications were incorporated in the development of the portal:

- Documentum 4.0
- Entrust GetAccess 7.0
- Microsoft Internet Information Services 5.0
- Microsoft SQL Server 2000
- Microsoft Windows 2000 Server
- Microsoft Windows 2000 Advanced Server
- Oracle 9.0
- Plumtree 4.5 Corporate Portal

3. http://buildsecurityin.us-cert.gov/bsi/about_us/authors/197-BSI.html (Lavenhar, Steven)

Plumtree's gadget architecture was utilized to allow the portal to provide access to a wide variety of applications and content. Gadgets were used for integrating legacy data and access to COTS applications such as SAP, Lotus Domino, Documentum, and custom programs. Single sign-on capabilities were provided by GetAccess.

Configuration Management

Change, configuration, and metadata management are critical components of a comprehensive life-cycle management strategy. Change management applications provide the tools for handling application and system modifications through pre-defined processes, minimizing the impact of change on the organization. Configuration and metadata management enables the sharing of information across components to allow for impact analysis, component reuse, common definitions, standards enforcement, and governance. CCC/Harvest was used as the solution for distributed change and configuration management.

The company developed a series of general controls that were applicable to all of it's IT systems. They were deemed essential best practices for ensuring the integrity, reliability, and quality of the systems. An intranet document repository contained up-to-date versions of all key documentation, including the latest policies and procedures. Shown below are some examples of the general controls that were in place:

- Antivirus policies
- Authentication/access controls
- Backup and recovery procedures
- Change management policies and procedures
- Coding standards and code review policies
- Disaster recovery/backup policies and procedures
- Firewall/VPN policies
- Incident management policies and procedures
- Internet usage policies
- Intrusion prevention and detection policies
- Laptop/workstation security
- Password policies
- Physical access security
- Security policies and procedures
- Service-level agreement policies
- Source code/document version control procedures
- Technical support policies and procedures
- Testing procedures and policies
- Version management/release management procedures

These general controls were standardized across the company, centrally administered, centrally controlled, and repeatable. Many additional best practices were in use. For example, all changes made to assets within the IT infrastructure were authorized, recorded, and reviewed before implementation. Adherence to the general controls was regularly assessed through design walkthroughs, code reviews, internal audits, and interviews to ensure that procedures and policies were properly followed. In addition to internal audits, external auditors from a well-respected security consulting firm were employed to provide assurance that the security controls in new systems were properly implemented. Also, periodic vulnerability assessments were performed including port scanning, checking for outdated software patches, checking for missing antivirus updates, and penetration tests.

Standard provisioning procedures were also in place to ensure that new users were assigned the correct privileges. These profiles determined the default permissions assigned to a new user of a given type and also enforced rules that prevented users from being assigned the wrong privileges. The principles of separation

of duties, least privilege, and user provisioning were strictly applied to all users of any system in use by the organization.

Patch Management Problems and Security Vulnerabilities

The organization had developed and tested detailed standards documentation that addressed the specific technologies in use by the corporation and established criteria by which compliance with the standards could be measured. While the standards documentation provided clear descriptions of the procedures needed to achieve compliance, there were some ambiguities with the process for achieving standards compliance. This was particularly true in the area of patch management.

Managing patches is an issue of critical importance to system administrators and IT managers. There have been several widely-publicized attacks and vulnerabilities related to patch management of Microsoft software. On January 24, 2003, the portal production environment was widely attacked by the Slammer Worm. The worm spread using a buffer overflow attack that exploited a flaw in Microsoft SQL Server 2000.

SQL Server 2000 has the ability to host multiple instances on a single physical machine. Each instance operates as though it was a separate server. The multiple instances cannot all use the standard SQL Server session port (TCP 1433). While the default instance listens on TCP port 1433, named instances of SQL Server listen on any port assigned to them. The SQL Server Resolution Service, which operates on UDP port 1434, provides a way for clients to query for the appropriate network endpoints to use for a particular SQL Server instance. Three security vulnerabilities resulted from flaws in the way this feature was implemented. The first two flaws are buffer overflows. By sending a carefully crafted packet to the Resolution Service, an attacker could cause portions of system memory (the heap in one case, the stack in the other) to be overwritten. Overwriting it with random data would result in the failure of the SQL Server service; overwriting it with carefully selected data would allow the attacker to run code in the security context of the SQL Server service. The third vulnerability was a denial of service vulnerability. SQL uses a keep-alive mechanism to distinguish between active and passive instances. It is possible to create a keep-alive packet that, when sent to the Resolution Service, will cause SQL Server 2000 to respond with the same information. An attacker who created such a packet, spoofed the source address so that it appeared to come from a SQL Server 2000 system, and sent it to a neighboring SQL Server 2000 system, could cause the two systems to enter a never-ending cycle of keep-alive packet exchanges. This would consume resources on both systems, slowing performance considerably [Microsoft 2003[47]].

Many organizations with proactive security patch management in place were not affected by these attacks, because they acted on information that Microsoft made available in advance of the attack. Due to the critical nature of the patch, it was deployed in the staging and production environments after a short pre-deployment test. After the patch was deployed security tests were performed in the staging environment. However, security tests were not performed in the production environment, which was a critical mistake. Since the staging and production environments were mirrored, it was assumed that the two environments were identical and that additional testing of the production environment was not necessary. However, there were subtle differences in the two environments. Specifically, the group security policies were not exactly identical, with the group policy in the production environment more restrictive. While the security patch was successfully deployed in all of the SQL Server machines in the staging environment, the deployment failed in the production environment due to the group security policy settings. The use of CCC/Harvest to deploy the patches and installation scripts to the servers compounded the problem because CCC/Harvest was not designed to provide the patch management functionality the company tried to implement.

Historically, Microsoft product teams have been free to develop their own technologies and engineering processes. Architectural differences among Microsoft products can in some cases necessitate differences in the way patches must be installed. A number of different technologies have been developed for installing patches, each associated with one or more products. However, each has unique characteristics that an administrator may need to be aware of in order to use it effectively. Each installer technology used by Microsoft has a unique set of numeric codes that it uses to indicate whether the installation

47. #dsy206-BSI_ms03

completed successfully. Likewise, installer technologies vary in their use of log files that provide more verbose diagnostic information. A best practice would be to establish a uniform set of return codes for troubleshooting problems with patch installations, along with procedures for examining installation log files for the patch installer program. However, this was not done because CCC/Harvest, which is a configuration management tool, was misapplied by the company. It was not designed to support patch management.

At approximately 9:30 P.M. Pacific Time on January 24, 2003, the SQL Slammer worm caused a dramatic increase in network traffic worldwide. A post-mortem analysis of the SQL Slammer worm shows that:

- The worm required roughly 10 minutes to spread worldwide, making it by far the fastest worm to date.
- In the early stages, the number of compromised hosts doubled in size every 8.5 seconds.
- At its peak, (achieved approximately three minutes after the worm was released), it scanned the net at over 55 million IP addresses per second.
- It infected at least 75,000 servers and probably many more.

Included in this large number of infected servers was every SQL Server machine in the company's production environment. In contrast, the staging environment, where the security patch was properly installed, was not affected.

Lessons Learned

This attack highlighted several important lessons about the nature of security vulnerabilities and patch management:

- Having an accurate understanding of the configuration of every computer in your environment is an important prerequisite for successful security patch management. Making assumptions, such as the synchronicity of the group policy settings in the staging and production environments, can lead to critical mistakes.
- Having an accurate understanding of every product and technology present in your environment is also an important prerequisite for successful security patch management.

Using CCC/Harvest for installing patches was a mistake. CCC/Harvest is an excellent configuration management tool that was misapplied as a patch management tool. Its use resulted in insufficient information about the successful installation of the necessary security patch in the production environment. This was not a technological failure but rather a human failure due to the misapplication of a technology. After the Slammer worm attack occurred, the company implemented Microsoft's Systems Management Server for patch management, and subsequent patch-related security problems have not occurred.

- Never make assumptions about the synchronization of different environments. Given the complexity of most enterprise architectures, subtle differences may have significant security implications.
- Deploying a security patch once may not be sufficient to eliminate a vulnerability. Installation logs should be checked rigorously after deployment, and thorough testing should be performed on each environment where a patch is installed. Regular scanning to identify the recurrence of vulnerabilities is equally important.

Because patch management is designed to give an organization control over the software updates it deploys, any organization planning to patch its operational environment should have

- Effective operations, including people who understand their roles and responsibilities.
- Tools and technologies that are appropriate for effective patch management.
- Effective project management of patch management processes.

References

[Microsoft 2003] "Buffer Overruns in SQL Server 2000 Resolution Service Could Enable Code Execution." Microsoft Security Bulletin MS02-039[67] (2003).

[Microsoft 2007] "Standardizing the Patch Experience[68]." Microsoft TechNet, 2007.

Cigital, Inc. Copyright

The Build Security In (BSI) portal is sponsored by the U.S. Department of Homeland Security (DHS), National Cyber Security Division. The Software Engineering Institute (SEI) develops and operates BSI. DHS funding supports the publishing of all site content.

1. mailto:copyright@cigital.com